Second-order connected attribute filters using max-trees

Ouzounis, Georgios K.; Wilkinson, Michael H.F.

Published in:
Mathematical Morphology: 40 years on

SECOND-ORDER CONNECTED ATTRIBUTE FILTERS USING MAX-TREES

Georgios K. Ouzounis and Michael H. F. Wilkinson

Institute of Mathematics and Computing Science
University of Groningen
(georgios, michael)@cs.rug.nl

Abstract The work presented in this paper introduces a novel method for second-order connected attribute filtering using Max-Trees. The proposed scheme is generated in a recursive manner from two images, the original and a modified copy by an either extensive or an anti-extensive operator. The tree structure is shaped by the component hierarchy of the modified image while the node attributes are based on the connected components of the original image. Attribute filtering of second-order connected sets proceeds as in conventional Max-Trees with no further computational overhead.

Keywords: second-order connectivity, Max-Tree, attribute filters, clustering, partitioning

Introduction

The concept of second-order connectivity [6, 8] is a generalization of conventional connectivity summarizing two perceptual conditions known as *clustering* and *partitioning*. In brief, when clustering objects close enough to each other in morphological terms, are considered as a single entity, while when partitioning isolated object regions interconnected by thin elongated segments are handled as independent objects. The theoretic framework developed to formalize this [8, 1] defines the two cases by means of connected openings that consider the intersection of the original image with the generalized connectivity map. Extensions to a multi-scale approach employing a hierarchical representation of connectivity have also been made. Two examples are connectivity pyramids [2] and *Connectivity Tree* [9], which quantify how strongly or loosely objects or object regions are connected.

Algorithmic realizations of this framework originally suggested the use of generalized binary and gray-scale reconstruction operators [1] for recovering the object clusters or partitions. This introduced a family of filters based on topological object relations with width as the attribute criterion. Efficient al-

C. Ronse et al. (eds.), Mathematical Morphology: 40 Years On, 65–74.

gorithms for the more general class of gray-scale attribute filters using second-order connectivity have not yet been proposed. In this paper we will present a method based on Max-Trees [7].

Our method builds a hierarchical representation based on gray scale image pairs comprising the original image and a modified copy by an increasing and either extensive or anti-extensive operator. The algorithm, referred to as *Dual Input Max-Tree* is inspired by [7, 10] and demonstrates an efficient way of computation of generalized area openings . The results extend easily to other attribute filters.

A presentation of our method is given in this paper which is organized as follows: The first section gives a brief overview of the concept of connectivity and attribute filters. A short description of second-order connectivities follows in the second section where the two cases of clustering and partitioning are described in a connected opening form. A review of the Max-Tree algorithm is given in the third section complemented by a description of our implementation while results and conclusions are discussed in the fourth section.

1. Connectivity and Connected Filters

This section briefly outlines the concept of connectivity from the classical morphological prospective. For the purpose of this analysis we assume a universal (non-empty) set E and we denote by $\mathcal{P}(E)$ the collection of all subsets of E. A set X representing a binary image such that $X \subseteq E$ is said to be connected if it cannot be partitioned into two non-empty closed or opened sets. Expressing this using the notion of *connectivity classes*, Serra [8] derived the following definition:

DEFINITION 1 *A family $\mathcal{C} \subseteq \mathcal{P}(E)$ with E an arbitrary non-empty set, is called a connectivity class if it satisfies:*

1 $\emptyset \in \mathcal{C}$ *and* $\{x\} \in \mathcal{C}$ *for* $x \in E$,

2 if $C_i \in \mathcal{C}$ *with* $i = 1, ...I$ *and* $\bigcap_{i=1}^{N} C_i \neq \emptyset$, *then* $\bigcup_{i \in I} C_i \in \mathcal{C}$

where $\{x\}$ *denotes a singleton.*

The class $\mathcal{C}$ in this case defines the connectivity on E and any subset of $\mathcal{C}$ is called a *connected set* or a *connected component*.

Given the connected sets $C_x \in \mathcal{C}$ containing $x \in X$, the connected opening connected, opening Γ_x can be expressed as the union of all C_x:

$$\Gamma_x(X) = \bigcup \{C_x \in \mathcal{C} | x \in C_x \text{ and } C_x \subseteq X\} \tag{1}$$

With all sets C_x containing at least one point of X in their intersection, i.e. x, their union $\Gamma_x(X)$ is also connected. Furthermore $\forall x \notin X, \Gamma_x(X) = \emptyset$.

Attribute Filters

Binary attribute openings attribute filter, opening [3] are a subclass of connected filters incorporating an increasing criterion T. The increasingness of T implies that if a set A satisfies T then any set B such that $B \supseteq A$ satisfies T as well. Using T to accept or reject a connected set C involves a trivial opening Γ_T which returns C if T is satisfied and $\emptyset$ otherwise. Furthermore, $\Gamma_T(\emptyset) = \emptyset$. The binary attribute opening is defined as follows:

DEFINITION 2 *The binary attribute opening Γ^T of a set X with increasing criterion T is given by:*

$$\Gamma^T(X) = \bigcup_{x \in X} \Gamma_T(\Gamma_x(X)) \tag{2}$$

The binary attribute opening is equivalent to performing a trivial opening on all connected components in the image. Note that if T is non-increasing we have an attribute thinning rather that an attribute opening.

2. Second-Order Connectivity

The concept of second-order connectivity is briefly reviewed in this section by visiting two characteristic cases. The clustering and partitioning operators presented next, exploit the topological properties of image objects by modifying the underlying connectivity while preserving the original shape.

Clustering Based Connected Openings

The first case concerns groups of image objects that can be perceived as clusters of connected components if their relative distances are below a given threshold. In morphological terms this is verified by means of an increasing and extensive operator ψ_c which modifies the connectivity accordingly. Merged objects in the resulting connectivity class $\mathcal{C}^{\psi_c}$ define the morphology of the clusters.

DEFINITION 3 *Let ψ_c be an increasing and extensive operator that modifies the original connectivity from $\mathcal{C}$ to $\mathcal{C}^{\psi_c}$. The clustering based connected opening $\Gamma_x^{\psi_c}$ associated with the generalized connectivity class $\mathcal{C}^{\psi_c}$ is given by:*

$$\Gamma_x^{\psi_c}(X) = \begin{cases} \Gamma_x(\psi_c(X)) \bigcap X & \text{if } x \in X \\ \emptyset & \text{if } x \notin X \end{cases} \tag{3}$$

Thus $\Gamma_x^{\psi_c}$ extracts the connected components according to Γ_x in $\psi_c(X)$, rather than X, and then restricts the results to members of X [8, 1].

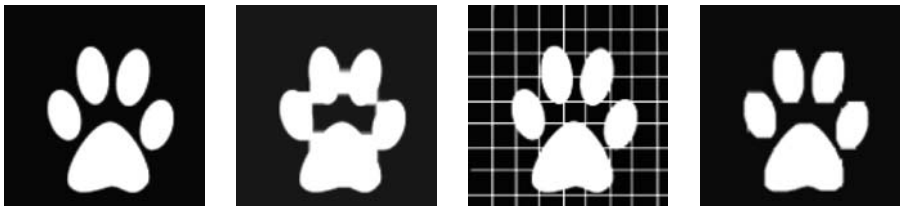

Figure 1. First pair: original image and the clustered connectivity map $\psi_c(X)$, Second pair: the original image in front of a grid (background) and the partitioned connectivity map $\psi_p(X)$.

Partitioning Based Connected Openings

Partitioning operators split wide object regions connected by narrow bridging segments which are often present due to image noise, background texture or out of focus details. The corresponding generalized binary connected opening extracts the intersection of the original image with the partitioned connectivity map $\psi_p(X)$, with ψ_p increasing and anti-extensive. To maintain the integrity of the original shape, all object level regions discarded by ψ_p are preserved as singletons in $\mathcal{C}$:

DEFINITION 4 *Let ψ_p be an increasing and anti-extensive operator that modifies the original connectivity from $\mathcal{C}$ to $\mathcal{C}^{\psi_p}$. The partitioning based connected opening $\Gamma_x^{\psi_p}$ associated with the generalized map $\mathcal{C}^{\psi_p}$ is given by:*

$$\Gamma_x^{\psi_p}(X) = \begin{cases} \Gamma_x(\psi_p(X)) \bigcap X & \text{if } x \in \psi_p(X) \\ \{x\} & \text{if } x \in X \setminus \psi_p(X) \\ \emptyset & \text{if } x \notin X \end{cases} \tag{4}$$

The problem that $X \setminus \psi_p(X)$ is fragmented into singletons is discussed in [11].

Second-Order Attribute Filters

Attribute filters as mentioned earlier apply a trivial opening Γ_T on the output of a binary connected opening Γ_x. Replacing Γ_x with a second-order connected opening Γ_x^{ψ} with ψ a generalizing operator (clustering or partitioning), gives rise to the concept of second-order attribute filters which in the binary case can be expressed as:

DEFINITION 5 *The binary second-order attribute opening of a generalized set $\psi(X)$ with increasing criterion T is given by:*

$$\Gamma_\psi^T(X) = \bigcup_{x \in X} \Gamma_T(\Gamma_x^\psi(X)) \tag{5}$$

The increasingness of these operators makes it possible to extend them directly to gray scale by threshold decomposition [4] of f, the mapping from the image

domain $\mathbf{M}$ to $\mathbb{R}$. Assuming that f can be decomposed to a set of binary images $T_h(f)$ resulting from thresholding f at all levels h, given by:

$$T_h(f) \;=\; \{x \in \mathbf{M} | f(x) \geq h\} \tag{6}$$

then superimposing them by taking their supremum leads to :

DEFINITION 6 *For a mapping $f : \mathbf{M} \rightarrow \mathbb{R}$, the gray scale second-order attribute opening $\gamma_\psi^T(f)$ is given by:*

$$(\gamma_\psi^T(f))(x) = \sup\{h | x \in \Gamma_\psi^T(T_h(f))\} \tag{7}$$

Thus, the second-order attribute opening of a gray scale image assigns each point of the original image the highest threshold at which it still belongs to a connected foreground component according to the second-order connectivity class $\mathcal{C}^\psi$. Other types of gray scale generalizations can be found in [7, 10].

3. The Max-Tree Algorithm

The Max-Tree is a hierarchical image representation algorithm introduced by Salembier [7] in the context of anti-extensive attribute filtering. The tree structure reflects the connected component hierarchy obtained by threshold decomposition of the given image with nodes and leaves corresponding to *peak components* and *regional maxima* respectively. A peak component P_h at level h is a connected component of the thresholded image $T_h(f)$ while a regional maximum M_h at level h is a level component no members of which have neighbors of intensity larger that h. A Max-Tree node C_h^k (k is the node index) corresponding to a certain peak component contains only those pixels in P_h^k which have gray-level h. In addition each node except for the root, points towards its parent $C_{h'}^{k'}$ with $h' < h$. The root node is defined at the minimum level $h_{\min}$ and represents the set of pixels belonging to the background.

Node attributes are parameters stored in the tree structure and are computed during the construction of the tree. In the case of the increasing attribute of node area the connected component k at level h inherits the area of all the peak components $P_{h'}^k$ connected to C_h^k at levels $h' > h$. Computing an area opening reduces to removing all nodes with area smaller than the attribute criterion λ from the tree. Note that the node filtering is a separate stage from the computation of attributes and connected component analysis [7] therefore consumes only a short fraction of the total computation time. Extensions to other types of attributes are trivial [3, 5, 7, 10].

Construction Phase

Max-Trees are constructed in a recursive manner from data retrieved from a set of hierarchical queues. The queues are allocated at initialization in the

```
/* flood(h, thisAttribute) : Flooding function at level h         */
attribute = thisAttribute        /* accounts for child attributes */
while (not HQueue-empty(h))             /* First step: propagation */
{ p = HQueue-first(h)                   /* retrieve priority pixel */
  STATUS[p] = NumberOfNodes[h]          /* STATUS = the node index */
  for (every neighbor q of p)           /* process the neighbors   */
  {   if (STATUS[q] == "NotAnalyzed")
      {   HQueue-add(ORI[q],q)          /* add in the queue        */
          STATUS[q] = "InTheQueue"
          NodeAtLevel[ORI[q]] = TRUE    /* confirm node existance  */
          if (ORI[q] > ORI[p])          /* check for child nodes   */
          {   m = ORI[q]
              child_attribute = 0
              do{                       /* recursive child flood   */
                  m = flood(m,child_attribute)
              } while (m != h)
              attribute += child_attribute   }}}}
NumberOfNodes = NumberOfNodes[h] + 1    /* update the node index   */
m = h-1                                 /* 2nd step: defines father*/
while ((m >= 0) and (NodeAtLevel[m] = FALSE))
   m = m-1
if (m >= 0){
   i = NumberOfNodes[h] - 1; j = NumberOfNodes[m];
} else
   The node C_i at level h has no father, i.e. its the root node
NodeAtLevel[h] = FALSE; node->Attribute = attribute;
node->Status = Finalized; thisAttribute = attribute;
return (m)
```

Figure 2. The flooding function of Salembier's algorithm adopted for area openings. The parameters `h` and `m` are the current and child node gray levels while `attribute` is a pixel count at level `h` within the same connected component. The parameter `thisAttribute` is used to pass child areas to parent nodes.

form of a static array called *HQueue* segmented to a number of entries equal to the number of gray levels. Data are accessed and stored in a first in - first out approach by the main routine (Fig. 2) which re-assigns priority pixels to the Max-Tree structure and stores new pixels retrieved from the neighborhood of the one under study, to the appropriate entries. The Max-Tree structure consists of nodes corresponding to pixels of a given peak component P_h^k at level h. Each node is characterized by its level h and index k and contains information about its parent node id, the node status and the attribute value.

The two structures are managed with the aid of three arrays; the *STATUS*[p], the *NumberOfNodes*[h] and the *NodeAtLevel*[h]. *STATUS* is an array of image size that keeps track of the pixel status. A pixel p can either be *NotAna-*

lyzed, *InTheQueue* or already assigned to node k at level h. In this case *STATUS*[p]=k. The *NumberOfNodes* is an array that stores the number of nodes created until that moment at level h. Last, *NodeAtLevel* is a boolean array that flags the presence of a node still being flooded at level h.

During initialization, the status of all image pixels is set to *NotAnalyzed*. Similarly the *NumberOfNodes* is set to zero while *NodeAtLevel* is set to *FALSE* for each gray level. After computing the image histogram, the *HQueue* and Max-Tree structures are allocated accordingly while the first pixel at level $h_{\min}$ is retrieved and placed in the appropriate queue. This pixel defines the root node and is passed on to the main routine (flood) as the initial parameter.

The flooding routine is a recursive function involved in the construction phase of the Max-Tree. It is initiated by accessing the first root pixel from the queue at level $h_{\min}$ and proceeds with flooding nodes along the different root paths that emerge during this process. The pseudo-code in Fig. 2 describes in detail the steps involved. Note that *ORI* is an image-size array that stores the pixel intensities. The construction phase terminates when all pixels have been assigned to their corresponding nodes and the Max-Tree structure is complete.

Constructing the Dual Input Max-Tree

Our implementation of the construction phase requires two input images. The first is the original image while the second is a copy modified by a clustering or partitioning operator. The idea can be summarized as follows; image data are loaded on the *HQueue* structure from the modified image to be mapped on the Max-Tree which is shaped by the histogram of the original image.

Upon finalizing the initialization process with both histograms computed, $h_{\min}$ is retrieved from the modified connectivity map $\psi(X)$ and placed in the corresponding queue while the three arrays are updated. The flooding function proceeds as described earlier by inspecting the neighbors of the starting pixel and distributes them to the appropriate queues. Within the *while* loop of Fig. 2 we add a test condition which checks for an intensity mismatch between the same pixel in the two images (see Fig. 3). Denoting with *P_ORI*the array storing the pixel intensity in the modified image if *ORI*[p] < *P_ORI*[p] where p is the pixel under study, the modified image is a result of an extensive operator ψ_c while if the opposite is true it is due to an anti-extensive operator ψ_p (see Fig. 3).

The first case involving clustering implies that p is a background pixel in the original image therefore it is regarded as connected to the current active node at level *ORI*[p] through the connected component at level *P_ORI*[p]; i.e. it defines a peak component at level *ORI*[p] to which p in the modified image is connected. *NodeAtLevel*[*ORI*[p]] is set and the status of p is updated to the node id at level *ORI*[p]. Additionally the node area is increased by a unit.

```
/* flood(h, thisAttribute) : Flooding function at level h         */
attribute = thisAttribute + node->Attribute /* node->Attribute is */
 /* added to account for pixels found during other calls to flood */
while (not HQueue-empty(h))          /* First step: propagation   */
{ p = HQueue-first(h)                /* retrieve priority pixel   */
  STATUS[p] = NumberOfNodes[h]       /* STATUS = the node index   */
  if(ORI[p]!=h){                     /* Detect intensity mismatch */
      NodeAtLevel[ORI[p]]=TRUE       /* Same for both cases       */
      node = Tree + NodeOffsetAtLevel[ORI[p]] + NumberOfNodes[ORI[p]]
      node->Attribute ++
      if(ORI[p]>h){                  /* Anti-extensive case       */
          node->Parent = NodeOffsetAtLevel[h] + NumberOfNodes[h]
          node->Status = Finalized; node->Level  = ORI[p]
          NumberOfNodes[ORI[p]] += 1; NodeAtLevel[ORI[p]] = FALSE
          attribute++ }          /* Finalizing the singleton node */
} else
      attribute++ /* If pixel intensity is the same in both images*/
/* The rest as in Figure 2 ...                                     */
return (m)
```

Figure 3. The flooding function of the *Dual Input Max-Tree* algorithm.

In the case where partitioning is involved the detected mismatch between the same pixel in the two images is of the form $P\_ORI[p] < ORI[p]$. Pixel p is therefore part of a discarded component according to Definition 4, and consequently is treated as a singleton. Singletons define a node of unit area at level $ORI[p]$ hence upon detection the node must be finalized before retrieving the next priority pixel from the corresponding queue at level $P\_ORI[p]$. This involves setting the node status to the node index at level $ORI[p]$ and detecting the parent node id. The area is simply set to a unit and upon completion $NodeAtLevel[ORI[p]]$ is set to *FALSE* indicating that this node is finalized.

The flooding function following this inspection proceeds with the neighboring pixels q updating the appropriate queues and setting the node flag for every pixel at the current level $P\_ORI[p]$. If a neighbor with a higher level $P\_ORI[q]$ is detected the process is halted at level $P\_ORI[p]$ and a recursive call to the flooding function initiates the same process at level $P\_ORI[q]$. This is repeated until reaching the regional maximum along the given root path. The Max-Tree structure is completed when all nodes are finalized.

Filtering

Once the Max-Tree structure is computed, filtering which forms a separate stage, is performed in a same way for both cases. Filtering the nodes based on the attribute value λ involves visiting all nodes of the tree once. If the node

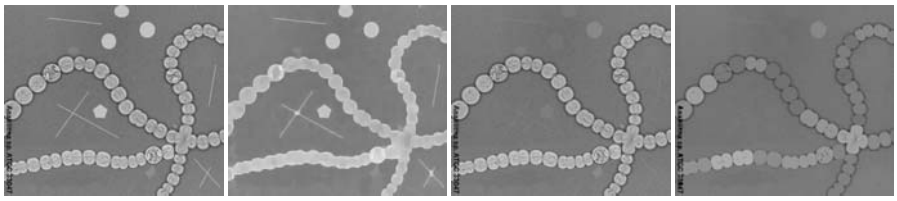

Figure 4. *Anabaena* colony (left to right): original image; connectivity map using closing by disc of radius 4; area opening ($\lambda = 900$) with Dual Input Max-Tree; area opening (same λ) with conventional Max-Tree. Image size 459×400 pixels.

attribute is less than λ the output gray level is set to that of the parent node and the comparison is repeated until the criterion is satisfied. The output image *Out* is generated by visiting all pixels p, retrieving their node ids from $ORI[p]$ and $STATUS[p]$ and assigning the output gray level of that node to $Out[p]$.

4. Discussion

The performance of the proposed algorithm was evaluated conducting a series of comparative experiments between the conventional and the dual input Max-Tree on sets of TEM images of bacteria. To demonstrate our results we chose two cases: one for clustering and one for partitioning. For a more extensive discussion of the utility of these filters the reader is refered to [1].

The first case involving clustering is demonstrated in Fig. 4 where artificial objects were added on an *Anabaena* colony to verify the filter's capability. Using a conventional Max-Tree representation, the attribute filter aiming at these additional objects, removes every connected component of area below the chosen criterion λ (set to 900) which includes parts of the colony too. In contrast to this, the same filter operated on the dual input Max-Tree considers the colony as a single object (as represented in the clustered connectivity map) and therefore removes only the unwanted objects of area less than λ. The second case considers partitioned objects and is demonstrated in Fig. 5. *Escherichia coli* cells in the original image are linked by filaments. Attribute filters on the conventional Max-Tree representation simply lower the intensity of these segments without eliminating the bridging effect. In the dual input Max-Tree however, the same object regions removed in the partitioned connectivity map (second from left), are converted to singletons in the original image hence any area filter with λ greater than the unit area discards them. This means that in the purely partitioning case, second-order connected attribute openings are equivalent to attribute openings of the connectivity map, as proven in [11].

The computational efficiency of our implementation has minimal difference from the conventional Max-Tree and this is due to loading two images and computing two histograms instead of one. The major computation takes place

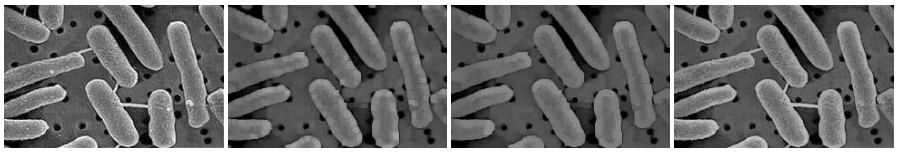

Figure 5. *E. coli* (left to right): original image; connectivity map ontained by opening with disc of radius 2; area opening using the dual input Max-Tree ($\lambda = 100$); area opening using conventional Max-Tree (same λ). Image size 242×158 pixels.

within the flooding function which differs from the original implementation in the two test conditions that verify the type of generalization. In both cases the same number of input pixels have to mapped into the same size Max-Tree structure therefore if the same image is used twice our algorithm performs as a conventional Max-Tree. Our flooding function (Fig. 3) uses a single routine to handle both cases of generalization. This is primarily motivated by our current investigation on increasing operators that are neither extensive nor anti-extensive and the potential to manage image pairs in which the modified connectivity map comprises sets of both clustered and partitioned objects. We are studying the properties of such more general second-order connectivities.

References

[1] U. Braga-Neto and J. Goutsias. Connectivity on complete lattices: New results. *Comp. Vis. Image Understand.*, 85:22–53, 2002.

[2] U. Braga-Neto and J. Goutsias. A multiscale approach to connectivity. *Comp. Vis. Image Understand.*, 89:70–107, 2003.

[3] E. J. Breen and R. Jones. Attribute openings, thinnings and granulometries. *Comp. Vis. Image Understand.*, 64(3):377–389, 1996.

[4] P. Maragos and R. D. Ziff. Threshold superposition in morphological image analysis systems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(5), 1990.

[5] A. Meijster and M. H. F. Wilkinson. A comparison of algorithms for connected set openings and closings. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):484–494, 2002.

[6] C. Ronse. Openings: Main properties, and how to construct them. Technical report, Université Louis Pasteur, Strasbourg, 1990.

[7] P. Salembier, A. Oliveras, and L. Garrido. Anti-extensive connected operators for image and sequence processing. *IEEE Trans. Image Proc.*, 7:555–570, 1998.

[8] J. Serra. Connectivity on complete lattices. *Mathematical Imaging and Vision*, 9:231–251, 1998.

[9] C. S. Tzafestas and P. Maragos. Shape connectivity: Multiscale analysis and application to generalized granulometries. *J. Math. Imag. Vis.*, 17:109–129, 2002.

[10] E. R. Urbach and M. H. F. Wilkinson. Shape-only granulometries and grey-scale shape filters. In *Proc. Int. Symp. Math. Morphology (ISMM) 2002*, pages 305–314, 2002.

[11] M. H. F. Wilkinson. Attribute-space connected filters. In *Proc. Int. Symp. Math. Morphology (ISMM) 2005*, 18-20 Apr 2005. These proceedings, pp. 85–94.